



TITLE:

重み付きレコードを扱う B^* 木の
ページネーションの改良について
(情報の記憶と利用に関する理論的
研究)

AUTHOR(S):

有川, 節夫

CITATION:

有川, 節夫. 重み付きレコードを扱う B^* 木のページネーションの改良について (情報の記憶と利用に関する理論的研究). 数理解析研究所講究録 1981, 423: 116-127

ISSUE DATE:

1981-04

URL:

<http://hdl.handle.net/2433/102578>

RIGHT:

重み付きレコードを扱う B^* 木の ページネーションの改良について

九州大学理学部 有川節夫

McCreight [1] による可変長レコードを扱う B^* 木を、少し拡張して重み付きレコードを扱う B^* 木として捉え、ページネーションのアルゴリズムをこれまで知られていた $O(n \log n)$ 時間のものから $O(n)$ 時間のものへ改良したので、それについて述べる。同時に文献 [1] における B^* 木の定義について検討を行う。

1. B 木

まず本稿で考える B 木について説明しよう。 K を全順序 ($<$) 集合とする。 K の要素をキーと呼ぶ。キー x には関連情報 α (番地や重みなど) が対応している。その組 $\gamma = (x, \alpha)$ をレコードといい、レコードの有限集合をファイルという。(ディスク上の) 番地付けのされた一定長の連続した領域をページという。以下では下図のような構造をしたページを考

p_0	(x_1, α_1)	p_1	(x_2, α_2)	p_2	//	p_{l-1}	(x_l, α_l)	p_l	あり
-------	-------------------	-------	-------------------	-------	----	-----------	-------------------	-------	----

えることにする。図において, p_i は他のページへのポインタであり, (x_i, α_i) はレコードである。

定義 1. 上のような構造をしたページを節 (node) とし
てもち, 次の条件を満たす有向木をクラス $T(m, h)$ の B 木
という: (m は偶数)

- i) 根から各葉までのパスの長さは h である。
- ii) 根以外の節は $\frac{1}{2}m \sim m$ 個のレコードをもつ。
- iii) 葉でない節 P は P 内のレコードの個数を l とする
とき, $l+1$ 個の子供をもつ。
- iv) 節 P 内のレコードを $(x_1, \alpha_1), (x_2, \alpha_2), \dots, (x_l, \alpha_l)$
とするとき

$$x_1 < x_2 < \dots < x_l$$

である。

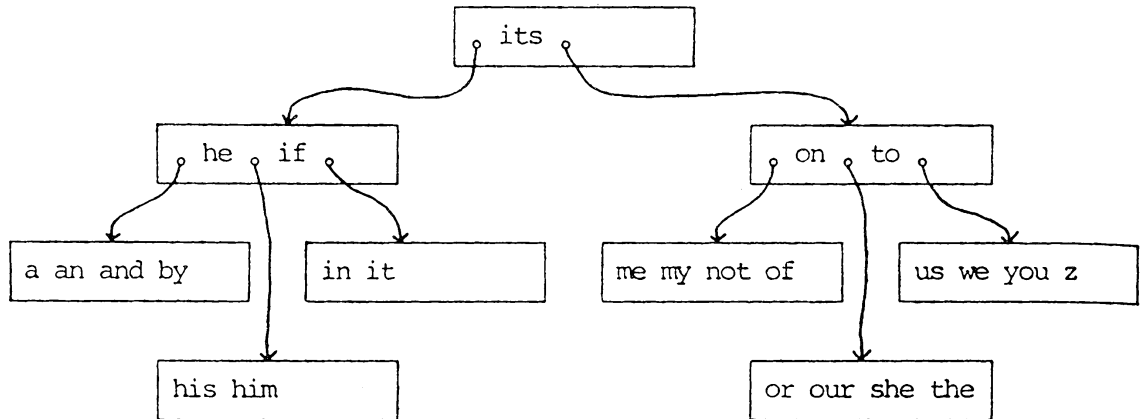
- v) $P(p_i)$ でポインタ p_i が指している節を表わし,
 $K(p_i)$ で $P(p_i)$ を根とする極大部分木の節内にあるキーの集
合を表わす。このとき次が成立する:

$$\forall y \in K(p_0), \quad y < x_1$$

$$\forall y \in K(p_i), \quad x_i < y < x_{i+1} \quad (1 \leq i < l)$$

$$\forall y \in K(p_l), \quad x_l < y.$$

<例> 下図は $(4, 3)$ の B 木の例である。キーの集合は図にある a から z までの英単語であり、順序は辞書式である。



なお図において、ポインタは矢印で示し関連情報は省略した。

2. 重み付きレコードを扱う B^* 木

レコードの挿入、削除の際にページを B 木の条件を満足するように再編成する必要が生じる。本稿では、重いレコードがなるべく B 木の根の近くにくるように再編成することを考える。

以下の議論に必要な約束を 2, 3 しておこう。 p_0, p_1, \dots, p_t をポインタ, $r_i = (\alpha_i, \alpha_i), \dots, r_t = (\alpha_t, \alpha_t)$ を $\alpha_i < \alpha_{i+1}$ ($1 \leq i < t$) なるレコードとする。このとき表現 $S = p_0 r_1 p_1 r_2 p_2 \dots p_{t-1} r_t p_t$ をスクロール (scroll) という。また t をその長さといい $|S|$ で示す。(B 木の各ページ P には長さ $1 \sim m$

のスクロールが記録されていることになる。そこで、以下では P でスクロールを表わすこともある。) レコード r とポイント p の組 (r, p) をエントリ (entry) と呼ぶ。レコード r_i の重みを $w(r_i)$ 又は単に $w(i)$ と書くことにする。また、 B 木の広がりを示す指数 m については $m = 2n$ として話を進める。

レコードの挿入、削除はページの分割、連結、ページ内容の詰めかえを伴うが、これは根ページまで伝播することもある。以下で述べる操作は、こうした伝播の各段階における方略である。対象になるページを P として、 P に対して $/$ からエントリ (s, p_s) が挿入/削除されるものとする。

定義 2. 以下の操作 (I) (II) により構成管理される B 木を B^* 木と呼ぶ。また (I) (II) の操作をページネーションと呼ぶ。

(I) レコードの挿入

P が満員 (full, $2n$ 個のレコードをもつ) でないときは、エントリ (s, p_s) を P の該当する場所に単に挿入するだけで済むから、以下では P が満員の場合だけを考えることにする。

1) P が根であるとき、 P にエントリ (s, p_s) を追加してできるスクロールを中央で分割して新しい根を作る。

2) P のすぐ下の弟ページ P' が満員でないとき； P にエ

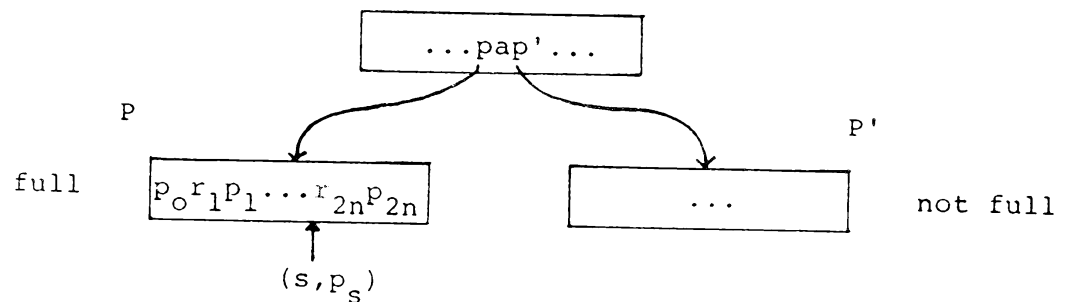
ントリ (s, p_s) を追加してできるスクロール, 親ページの境界レコード及び P' のスクロールを連結したスクロール

$$q_0 t_1 q_1 \cdots t_{2n+l+2} q_{2n+l+2} \quad (l = |p'|)$$

から, 条件

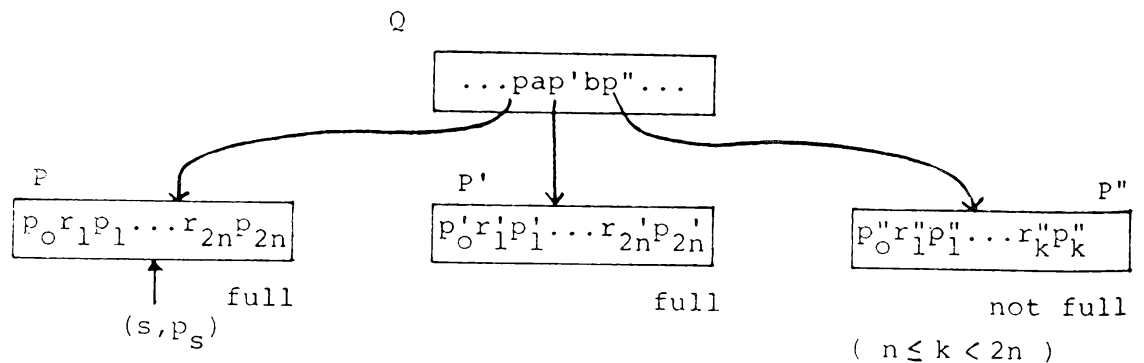
$$l+2 \leq x \leq n+l+2$$

の下で $W(t_x)$ を最大にする x を求め, $q_0 t_1 \cdots t_{x-1} q_{x-1}$ を P に残し, 親ページの境界レコード $q_{2n+2} (=a)$ を t_x で置き換え, $q_{x+1} t_{x+2} \cdots q_{2n+l+2}$ を P' に置く.



なお P のすぐ上の兄ページ P' が満員でないときも同様である.

3) P のすぐ下の弟が満員で, その次の弟が満員でない (下図) のとき;



エントリ (s, p_s) と P, P', P'' のスクロール及び Q の境界
レコード a, b を綴り合せてできるスクロール

$$q_0 t_1 q_1 \cdots t_{n+k+2} q_{n+k+2}$$

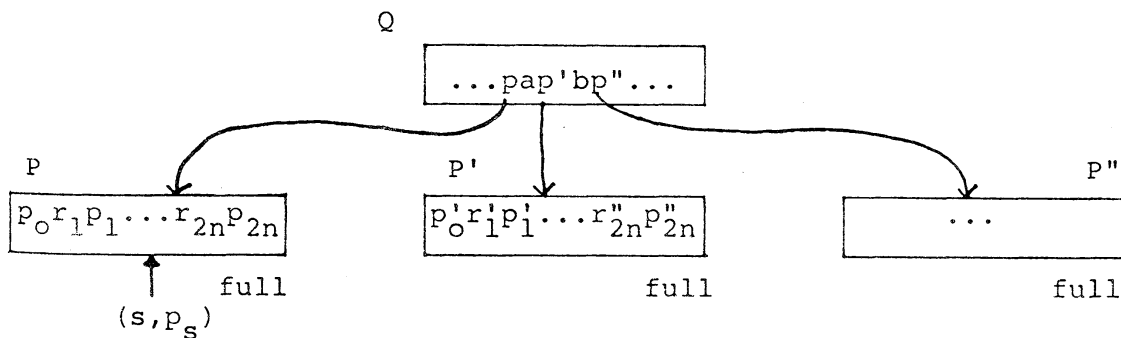
から, 条件

$$\begin{cases} n+1 \leq x \leq 2n+1 \\ x+n+1 \leq y \leq x+2n+1 \\ 2n+k+2 \leq y \leq 3n+k+2 \end{cases}$$

の下で $w(t_x) + w(t_y)$ を最大にする (x, y) を求め, 親ペ
ージの境界レコード a, b を t_x, t_y で置き換え, $q_0 t_1 \cdots$
 $t_{x-1} q_{x-1}$ を P へ, $q_x t_{x+1} \cdots t_{y-1} q_{y-1}$ を P' へ, $q_y t_{y+1}$
 $\cdots t_{n+k+2} q_{n+k+2}$ を P'' へ置く.

なお, P のすぐ上の兄が満員でその上の兄が満員でない
場合も同様である.

4) P のすぐ下の弟もその次の弟も満員であるとき;



エントリ (s, p_s) と P, P' のスクロール及び Q の境界レ
コード a を綴り合せてできるスクロール

$$q_0 t_1 q_1 \cdots t_{4n+2} q_{4n+2}$$

から, 条件

$$\begin{cases} n+1 \leq x \leq 2n+1 \\ x+n+1 \leq y \leq x+2n+1 \\ 2n+2 \leq y \leq 3n+2 \end{cases}$$

の下で $w(t_x) + w(t_y)$ を最大にする (x, y) を求め, 親ページの境界レコード a を t_y で置き換え, $q_0 t_1 \cdots t_{x-1} q_{x-1}$ を P へ, $q_x t_{x+1} \cdots t_{y-1} q_{y-1}$ を新しいページ P''' へ, $q_y t_{y+1} \cdots t_{4n+2} q_{4n+2}$ を P' へ置き, エントリ (x, p''') を親ページに渡す. (Q に次の段階で (x, p''') が挿入されることになる. 即ちこの操作は伝播することになる.)

なお, P のすぐ上の兄もその上の兄も満員である場合も同様である.

(II) レコードの削除

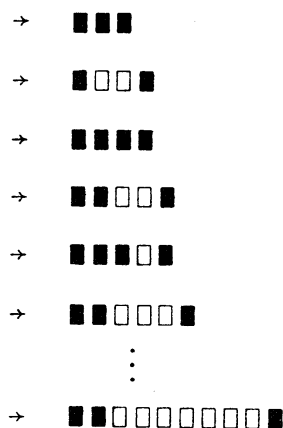
レコードの削除に伴ってページ P のレコードの個数が $\frac{2}{3}m (= \frac{4}{3}n)$ を割るときには, 高々2枚の隣接した (即ち, すぐ上の2人の兄, 又はすぐ下の2人の弟) ページを対象にして, ページ内容の詰め合せを試みる. もし不可能ならば, この3ページを対象に (I) の 3) に相当する割り振りを行う. ただし, この場合には $0 \leq k < 2n$ となる.

<注意> McCreight[1] では、可変長レコードを扱うB木について議論している。これは重み付きレコードの特別の場合である。また彼は B^* 木を、我々の定義1のii)を

ii)' 根以外の節は $\frac{1}{2}m \sim m$ 個のレコードをもち、しかも兄弟が3個以上の節からなる場合には、その兄弟は平均して $\frac{2}{3}m$ 個のレコードをもち、

という条件で置き換えたものに相当する定義をしている。しかも、ページネーションの操作については上記(I)(II)を使っている。しかし、(I)(II)の操作は彼のいう B^* 木の定義を満足しない。実際に、例はいくらでも作れるが簡単なものとして下図のような例がある。図において、 $\square \cdots \square$ は或る兄弟ページを表わし、 \square で $\frac{1}{2}$ full を \blacksquare で full を表わしている。図における最終段階でのページの使用率は、

$$\frac{3 + \frac{1}{2} \times 7}{10} = \frac{39}{60} < \frac{40}{60} = \frac{2}{3}$$



となり、彼の B^* 木の定義における条件 ii)' を満足しない。

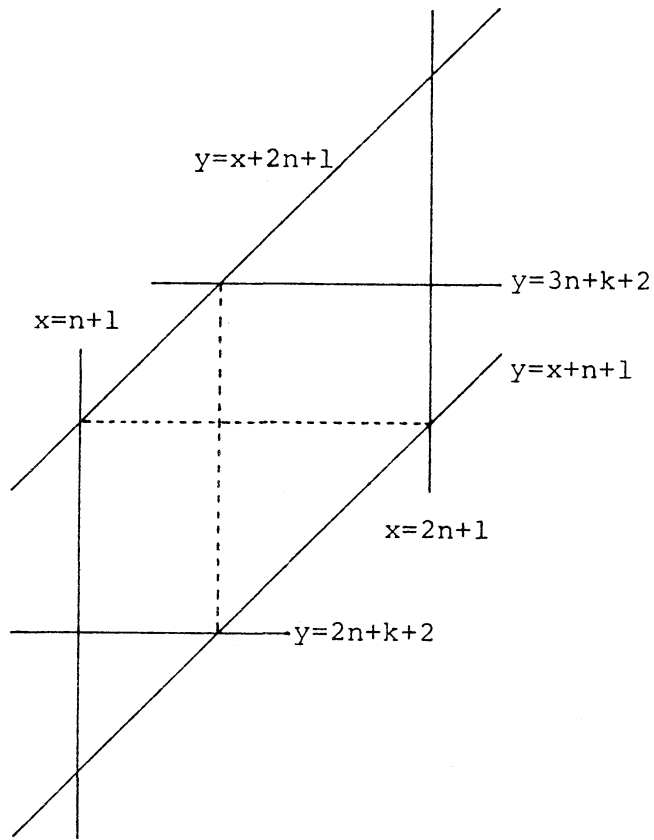
3. ページネーションのアルゴリズム

前節で述べたページネーションは挿入時における 3), 4) と削除時における 3) に相当する場合を除けば、 $O(n)$ 時間で実行できることは明らかである。3), 4) については素朴な方法では $O(n^2)$ 時間必要である。McCreight [1] は、ヒープ・ソートを使った $O(n \log n)$ 時間の方法を示しているが、簡単な工夫により、 $O(n)$ 時間にちぢめることが示せる。即ち、

定理 重み付きレコードを扱う B^* 木のページネーションは $O(n)$ 時間で可能である。

(証明) 挿入時における 3), 4) 及び削除時における 3) に相当する場合について考えれば十分である。これら 3 つのケースは $0 \leq k < 2n$ に対する 3) の場合としてまとめられる。そこで、3) における 3 つの条件式を x, y 座標に図示すると次のような六角形の領域が得られる。従って問題はこの領域(境界も含む)内の格子点で $w(x) + w(y)$ を最大にするような (x, y) を求める問題になる。この領域における $w(x) + w(y)$ の最大値は 2 つの矩形と 2 つの三角形における最大値の大きい方の値である。

矩形においては



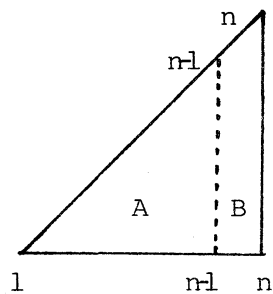
$$\max \{w(x) + w(y); a \leq x \leq b, c \leq y \leq d\}$$

$$= \max \{w(x); a \leq x \leq b\} + \max \{w(y); c \leq y \leq d\}$$

であるから、最大値を与える点 (x, y) は $O(n)$ 時間内で求められる。

一方三角形においては、この三角形を下図のように見る

と、



$$\begin{aligned}
& \max \{w(x) + w(y); (x, y) \in A \cup B\} \\
&= \max (\max \{w(x) + w(y); (x, y) \in A\}, \\
&\quad w(n) + \max (\max \{w(y); 1 \leq y \leq n-1\}, w(n)))
\end{aligned}$$

であるから、やはり $O(n)$ 時間で最大値を与える点 (x, y) を求めることができる。□

重み付きレコードを扱う B^* 木を導入し、そのページネーションについて単純で速いアルゴリズムを示した。この B^* 木は McCreight [1] の可変長レコードを扱う B^* 木の拡張になっていることは明らかであろう。ここで考えた重みとしてはレコードの使用頻度などがある。 B^* 木を定義するための順序の他に、頻度を考慮して、ページネーションにおける $1/2 \sim full$ で平均して $2/3$ という自由度を積極的に使って、使用頻度の高いレコードを B^* 木の根になるべく近いところに置こうというわけである。こうすると、外部記憶装置へのアクセス回数を少なくすることができる。

参考文献

- [1] McCreight, E.: Pagination of B^* -Trees with Variable-Length Records, CACM, 20 (1977), 670 - 674.
- [2] Bayer, R., and McCreight, E.: Organization and

Maintenance of Large Ordered Indexes, Acta Informatica, 1 (1972), 290 - 306.

- [3] Comer, D. : The Ubiquitous B-Tree, Computing Surveys, 11 (1979), 121 - 136.